RSAC 2025 Conference

Many Voices. One Community.

SESSION ID: CRYP-T09A

# Non-Interactive Threshold BBS+ From Pseudorandom Correlations

**Sebastian Faust, Carmit Hazay, David Kretzler, Leandro Rometsch, Benjamin Schlosser**

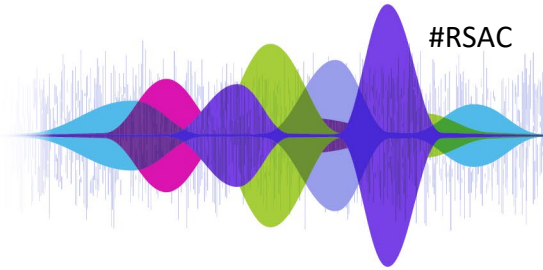
Technical University of Darmstadt
benjamin.schlosser@cac.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Bar-Ilan University
אוניברסיטת בר-אילן

ATHENE
Nationales Forschungszentrum für angewandte Cybersicherheit

CROSSING



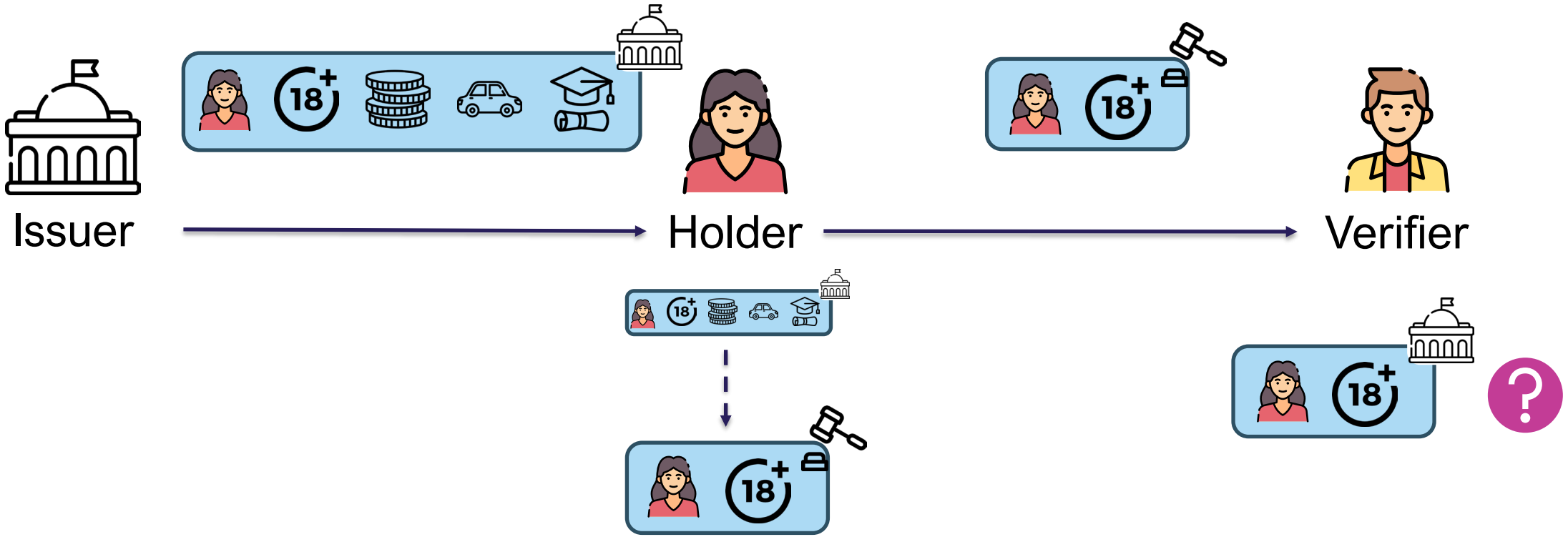
# Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

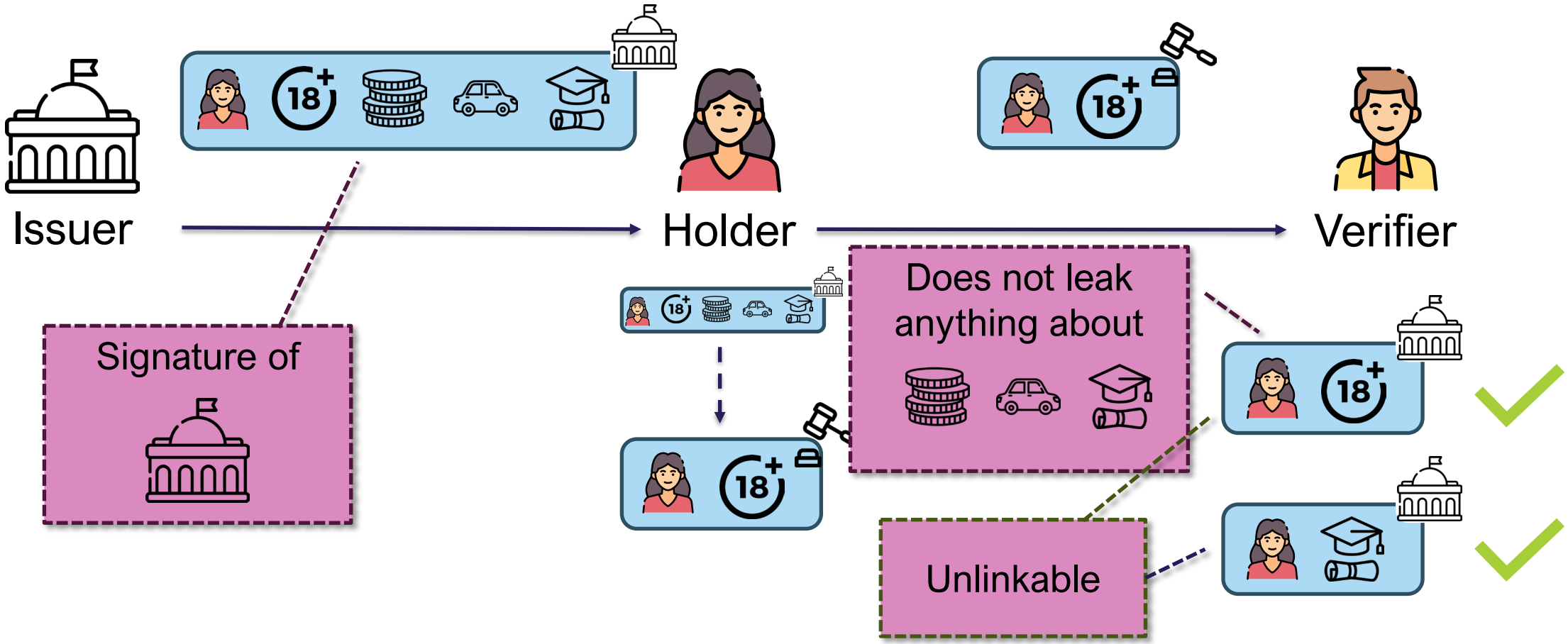
Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

# Big Picture: Verifiable Anonymous Credentials

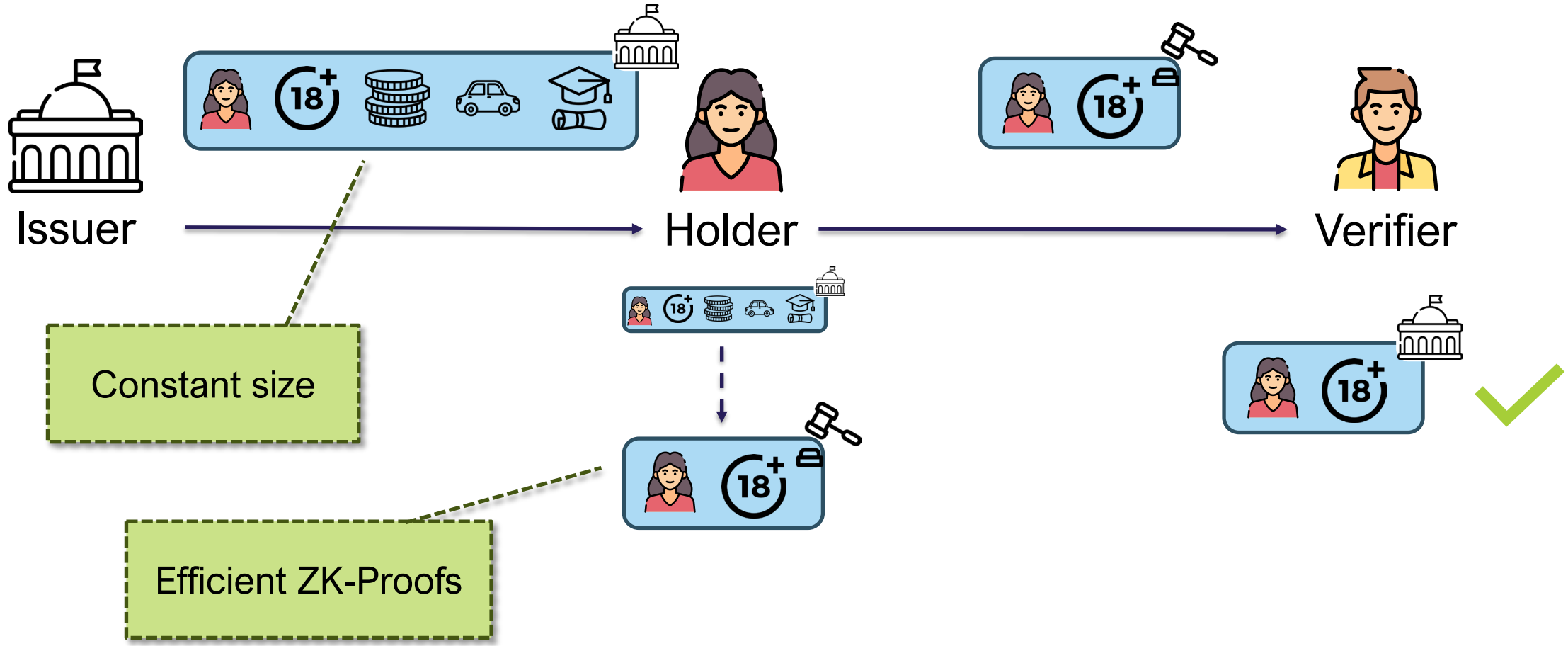


# Big Picture: Verifiable Anonymous Credentials



# Big Picture: Verifiable Anonymous Credentials

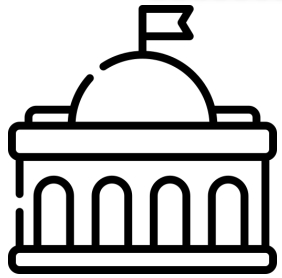
## Instantiation based on BBS+ Signatures



# Threshold Issuance

Replace centralized signing with distributed protocol

Adversary has to corrupt **t** out of **n** committee members



Issuer

Single Point of Failure

Increased robustness

Better availability

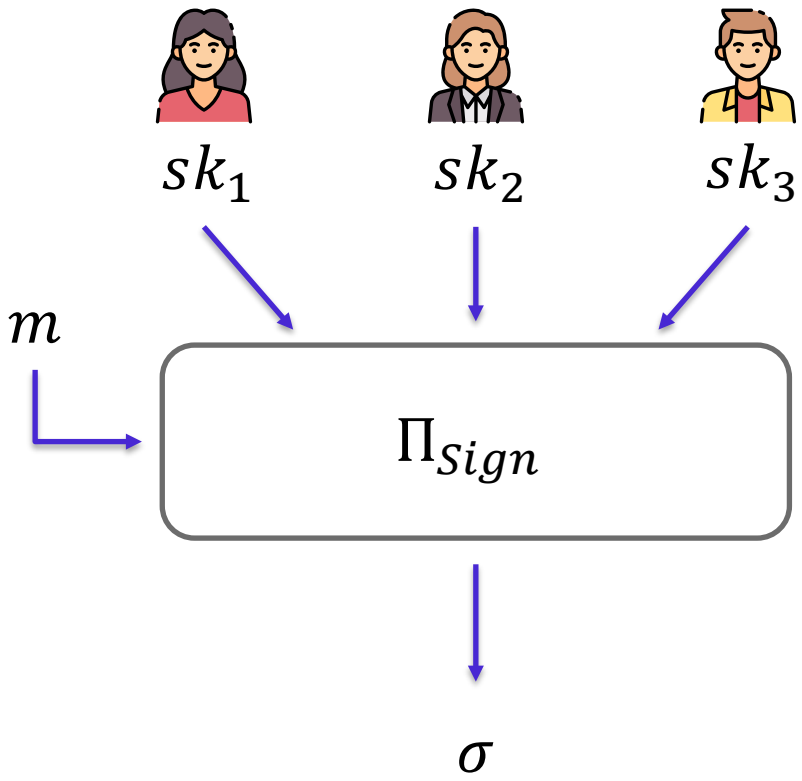
RSAC 2025 Conference

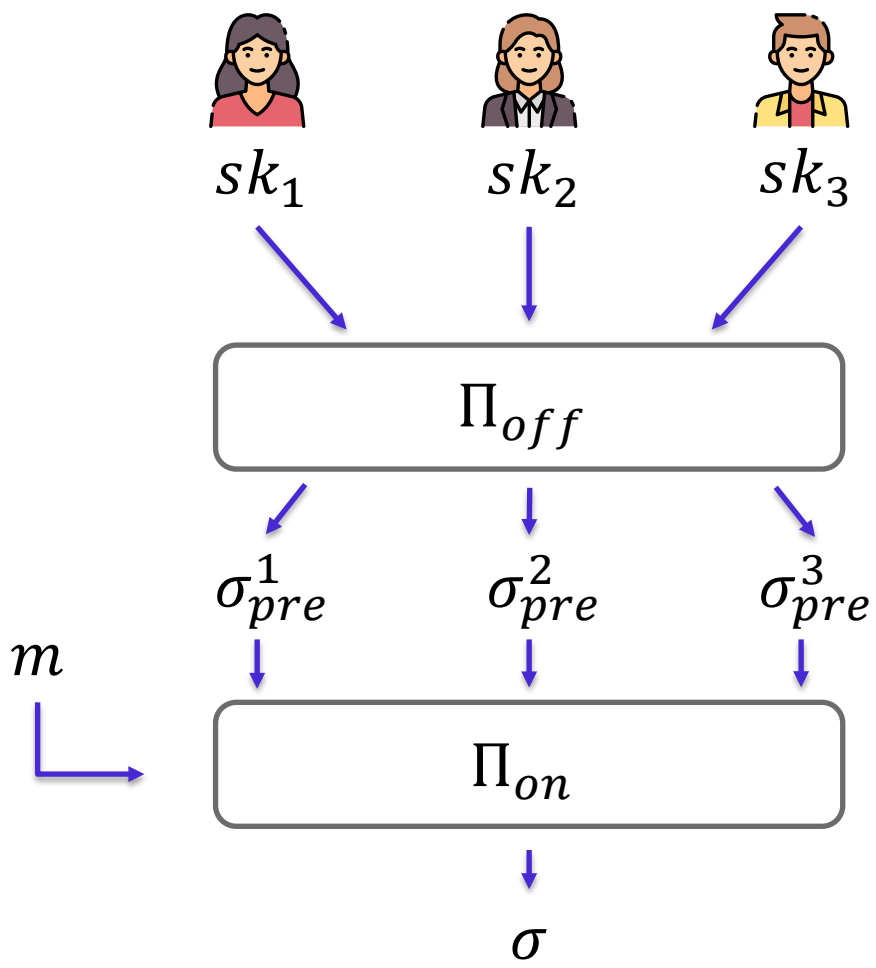
# Threshold BBS+ Protocol

## Non-Interactive; From Pseudorandom Correlation

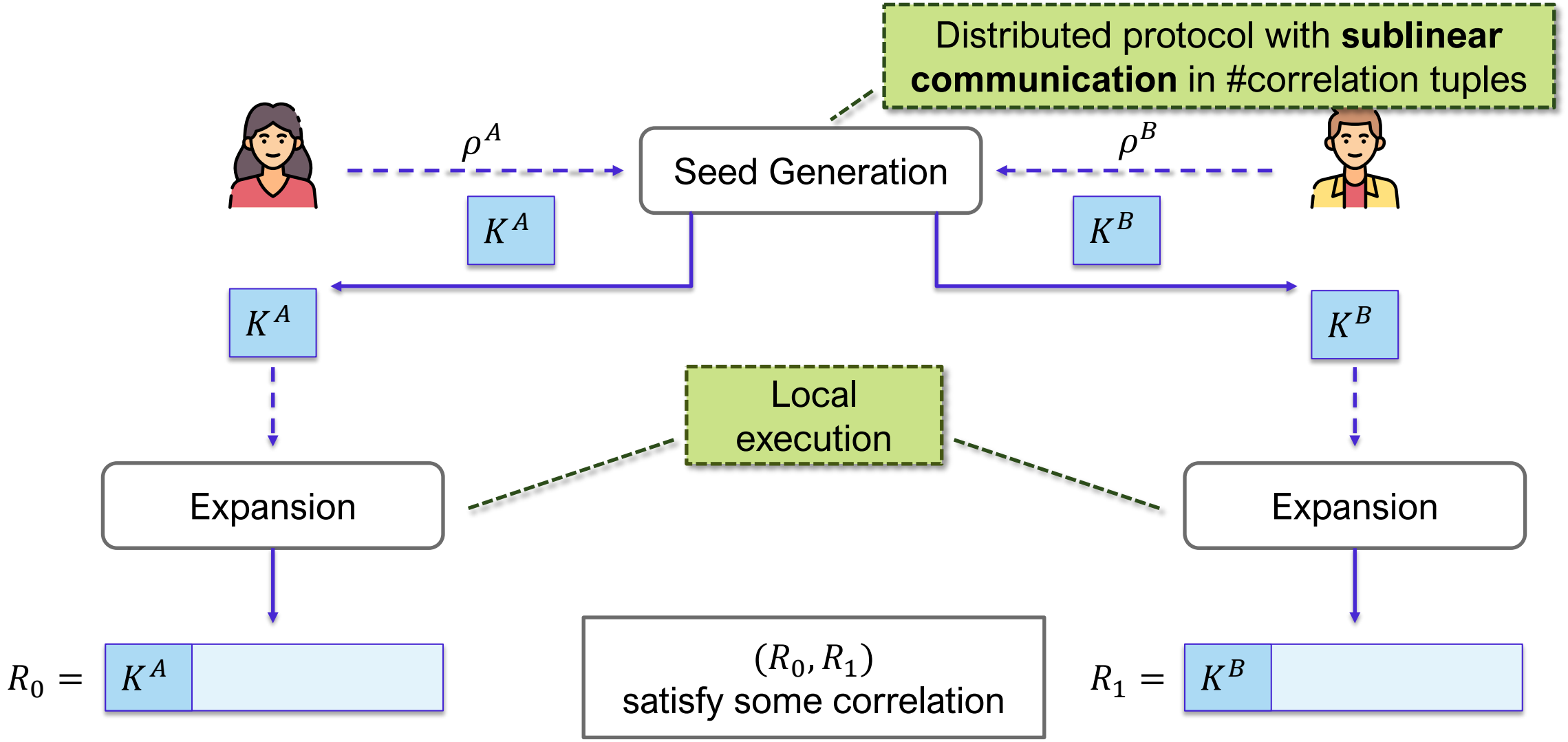
Many Voices.
**One Community.**

# Tool 1: Offline/Online Paradigm



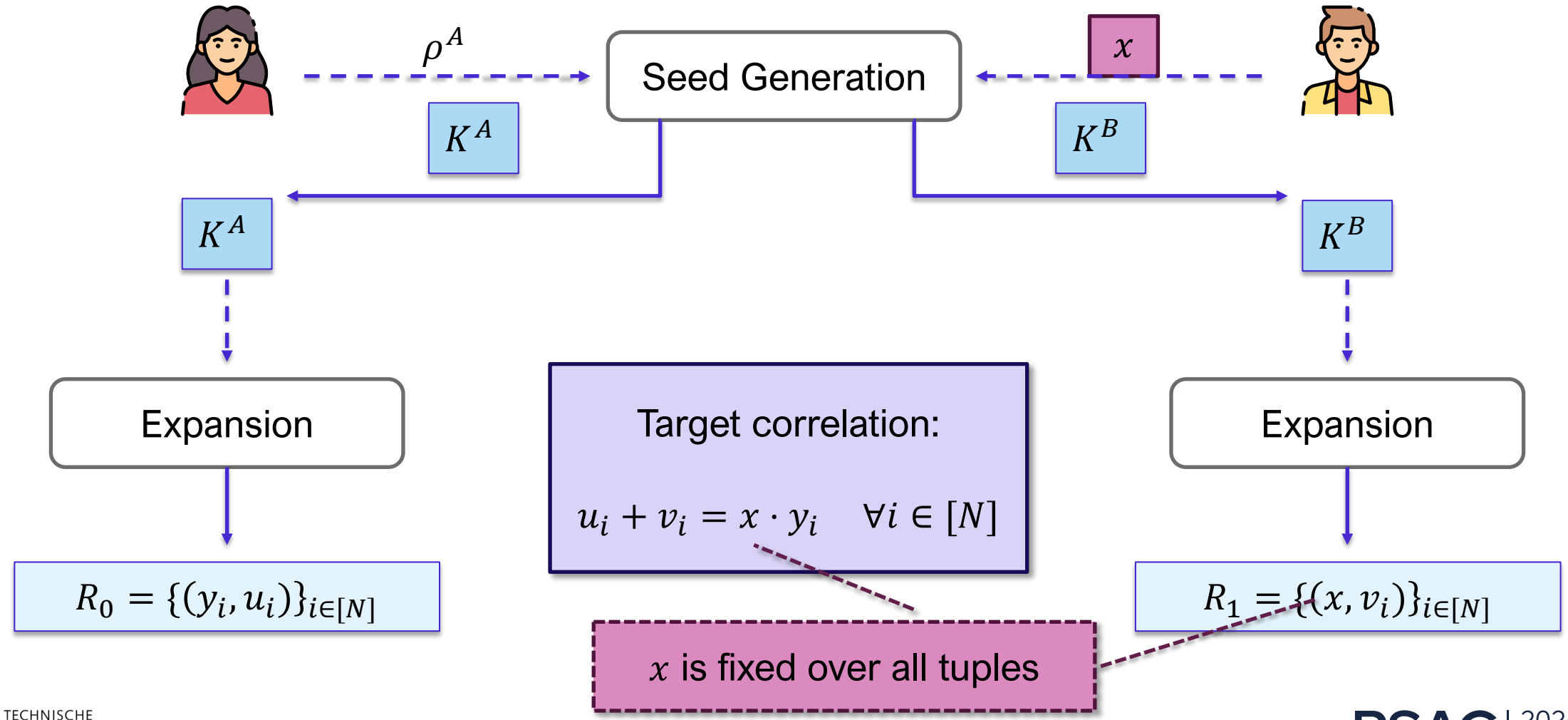


# Tool 2: Pseudorandom Correlation Generator

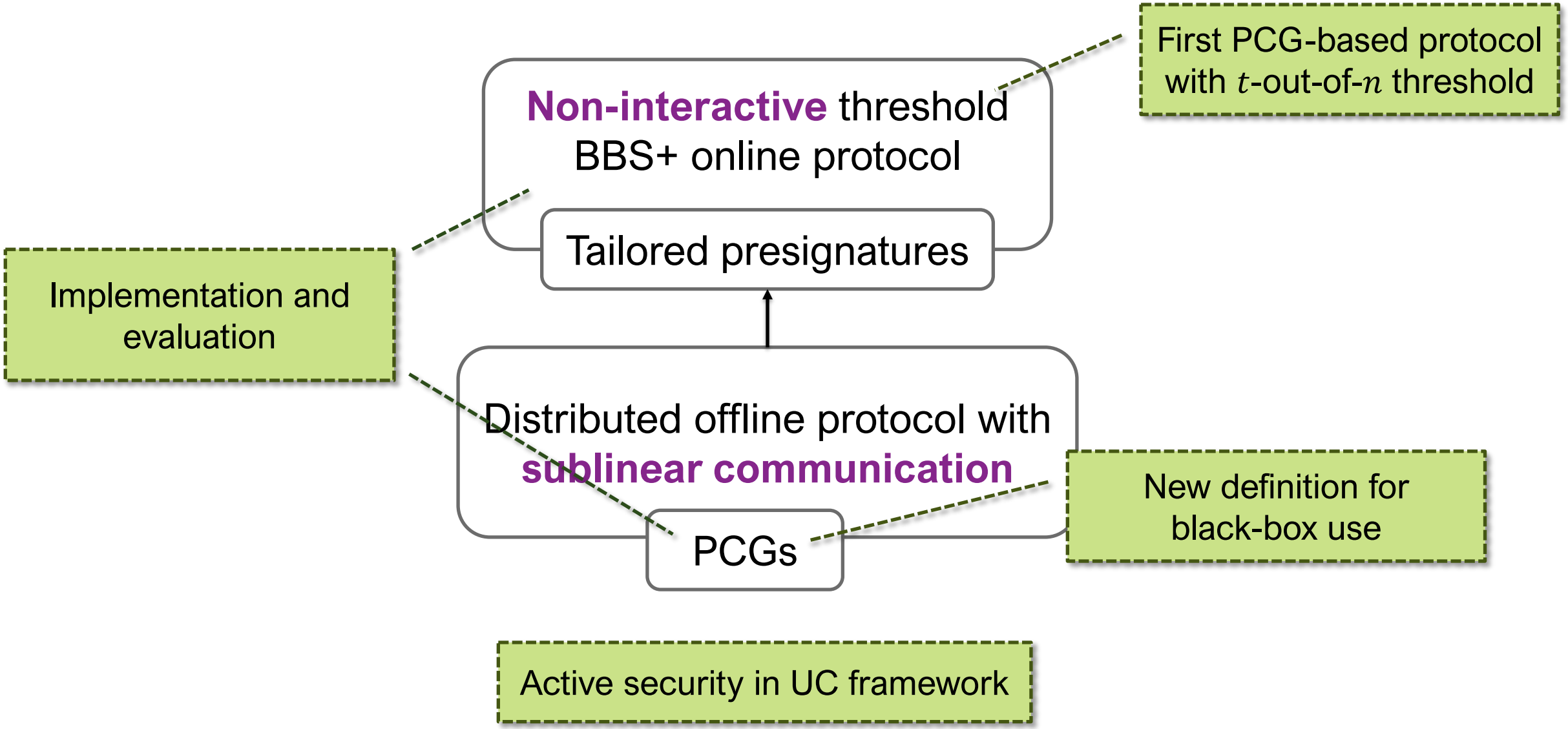


# Tool 2: PCG for VOLE

Vector Oblivious Linear Evaluation



# Our Contribution


First PCG-based protocol with $t$-out-of-$n$ threshold
Non-interactive threshold BBS+ online protocol
Tailored presignatures
Implementation and evaluation
Distributed offline protocol with sublinear communication
PCGs
New definition for black-box use
Active security in UC framework

RSAC 2025 Conference

# Non-Interactive Online Protocol

# BBS+ Signature

- Based on group signature scheme by Boneh, Boyen, and Shacham [BonehBS04]
- Transformed to signature scheme by [AuSMC13]
- Optimized by [TessaroZ23]

$$sk \xleftarrow{\$} \mathbb{Z}_p, \qquad pk = g^{sk}$$

$$\sigma = (A, e), \text{ where } e \xleftarrow{\$} \mathbb{Z}_p \text{ and}$$
$$A = \left(g_1 \cdot \Pi_{k\in[\ell]} h_\ell^{m_k}\right)^{\frac{1}{sk+e}}$$

# Roadmap to Threshold Protocol

For simplicity, we look at additive sharing next

$$sk \xleftarrow{\$} \mathbb{Z}_p, \qquad pk = g^{sk}$$

$$\sigma = (A, e), \text{ where } e \xleftarrow{\$} \mathbb{Z}_p \text{ and}$$

$$A = \underbrace{\left(g_1 \cdot \Pi_{k\in[\ell]} h_\ell^{m_k}\right)}_{B}^{\frac{1}{sk+e}}$$

1. Split secret key using Shamir's secret sharing

$sk_1$ $sk_2$ $sk_3$

2. Sample $e$ using Random Oracle

$$e \leftarrow H(\{m_\ell\}_{\ell\in[k]}, \mathcal{T})$$

3. Jointly compute

$$A = B^{\frac{1}{sk+e}}$$

Computing the inverse of $\mathrm{sk} + e$ in the exponent → non-linear operation

# From Presignatures to Signature

$sk_1$

$sk_2$

$sk_3$

Presignatures: $(a_1, \beta_1)$ $(a_2, \beta_2)$ $(a_3, \beta_3)$

$$\sum a_i = a \xleftarrow{\$} \mathbb{Z}_p$$

$$\sum \beta_i = a \cdot sk$$

Local computation:

$$A_i = B^{a_i}$$
$$\delta_i = \beta_i + a_i \cdot e$$

No interaction between signers

Reconstruction:

$$\epsilon = \left(\sum \delta_i\right)^{-1} = \frac{1}{a(sk+e)}$$

$$A = \left(\prod A_i\right)^{\epsilon} = (B^a)^{\frac{1}{a(sk+e)}} = B^{\frac{1}{sk+e}}$$

RSAC 2025 Conference

# Offline Protocol

## Computing Presignatures Using PCGs

Many Voices.
**One Community.**

# How to Compute Presignatures?

$$\sum a_\ell = a,$$
$$\sum \beta_\ell = a \cdot sk$$

$$(a_1 + a_2)(sk_1 + sk_2)$$

Share-wise multiplication

$$= a_1 \cdot sk_1 + \boldsymbol{a_1 \cdot sk_2} + \boldsymbol{a_2 \cdot sk_1} + a_2 \cdot sk_2$$

Multiplicative-to-additive conversion

$$= a_1 \cdot sk_1 + \boldsymbol{u_1} + \boldsymbol{v_2} + \boldsymbol{u_2} + \boldsymbol{v_1} + a_2 \cdot sk_2$$

Rearrange

$$= a_1 \cdot sk_1 + \boldsymbol{u_1} + \boldsymbol{v_1} + \boldsymbol{u_2} + \boldsymbol{v_2} + a_2 \cdot sk_2$$

$\boldsymbol{\beta_1}$ $\boldsymbol{\beta_2}$

# How to Compute Presignatures?

$$\sum a_\ell = a,$$
$$\sum \beta_\ell = a \cdot sk$$

$$(a_1 + a_2)(sk_1 + sk_2)$$

$$= a_1 \cdot sk_1 + \boldsymbol{a_1 \cdot sk_2} + a_2 \cdot sk_1 + a_2 \cdot sk_2$$

$$= a_1 \cdot sk_1 + \boldsymbol{u_1} + \boldsymbol{v_2} + u_2 + v_1 + a_2 \cdot sk_2$$

$$= a_1 \cdot sk_1 + u_1 + v_1 + u_2 + v_2 + a_2 \cdot sk_2$$

$\beta_1$ $\beta_2$

$\rho_2$ $sk_2$

VOLE

$(a_1, u_2)$ $(sk_2, v_1)$

$$u_1 + v_2 = a_1 \cdot sk_2$$

With sublinear communication using PCGs

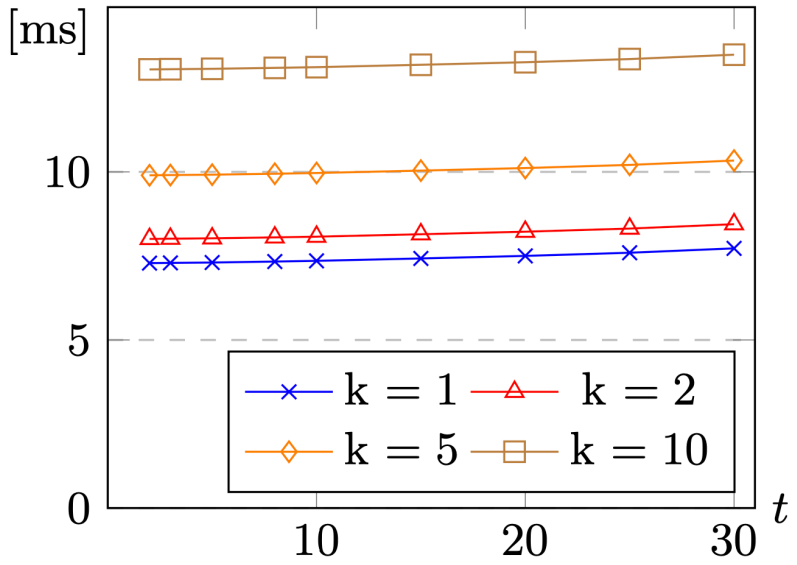
RSAC 2025 Conference

# Evaluation

# Evaluation – Online Protocol

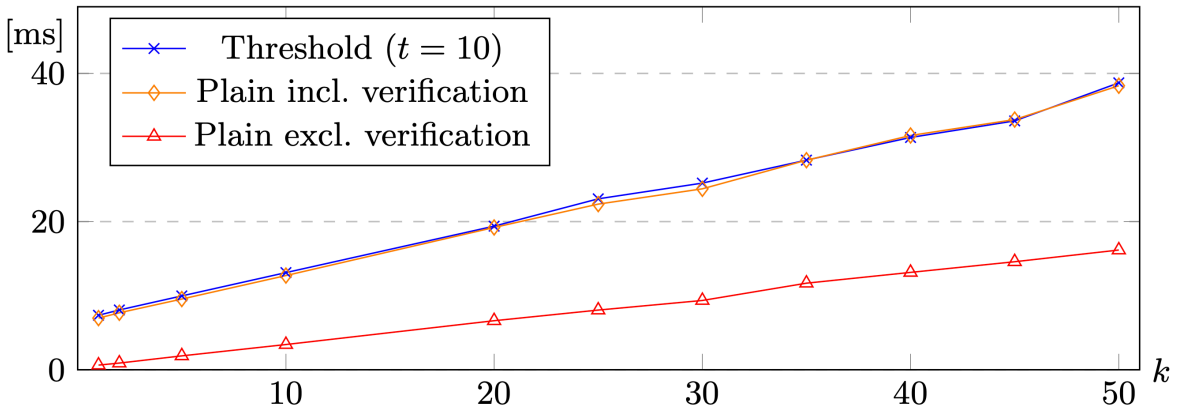
Implementation in Rust

## Total signing runtimes




- #servers is insignificant

## Comparison with non-threshold signing




- Threshold signing almost as fast as non-threshold signing + verification

Fast online protocol at the cost of preprocessing!

# Take-Home Messages

1. First PCG-based threshold signature protocol with

   **Non-interactive** online phase
   +
   offline phase with **sublinear communication**

2. Possible to move overhead of thresholdization almost completely to offline phase

**Open Questions:**

- Can we use PCGs for other threshold signature schemes?
- How to design more efficient **P**seudorandom **C**orrelation **F**unctions?

RSAC 2025 Conference

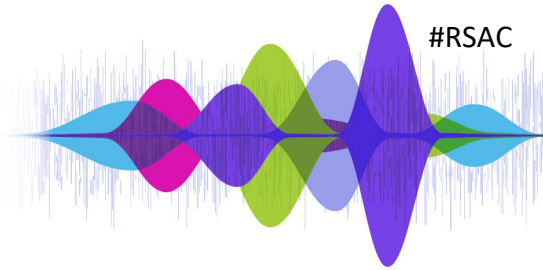
Many Voices. One Community.

SESSION ID: CRYP-T09B

# Updatable Public Key Encryption with Strong CCA Security: Security Analysis and Efficient Generic Construction

**Kyoichi Asano**[1] and Yohei Watanabe[1, 2]

1: The University of Electro-Communications, Tokyo, Japan
2: National Institute of Advanced Industrial Science and Technology, Tokyo, Japan



# Disclaimer

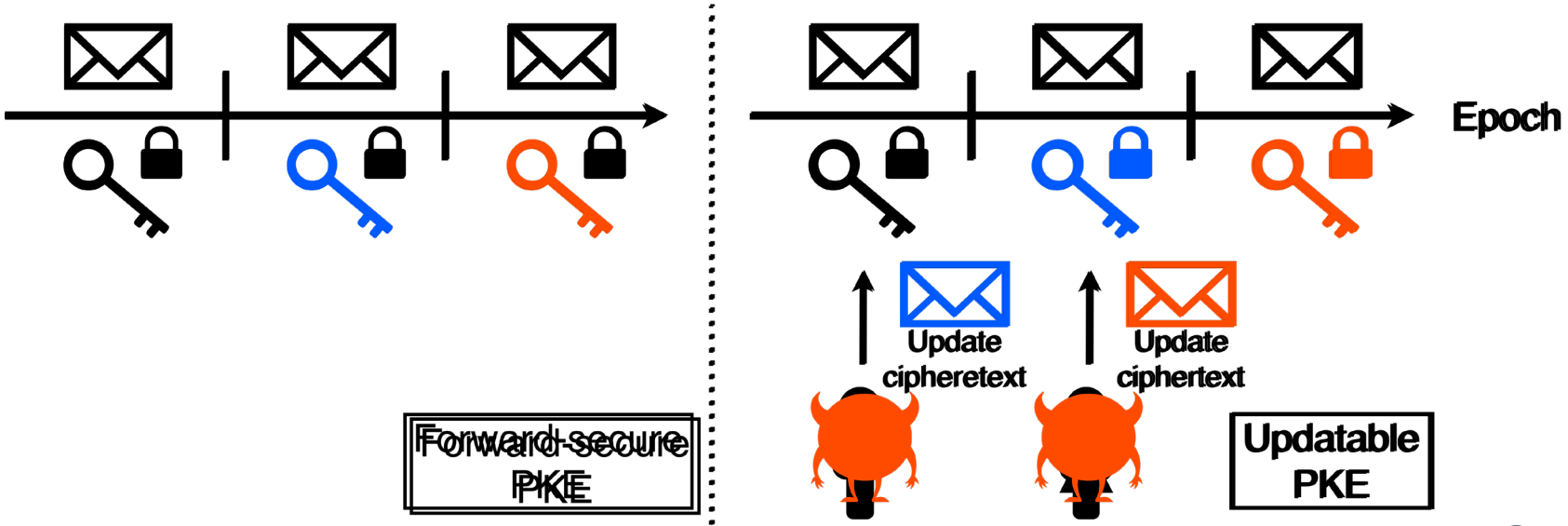
Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

# Updatable Public Key Encryption (UPKE) [AMM19, ACD+20]

- Intended for applications to secure messaging
- A natural relaxation of forward-secure public key encryption
  - Satisfy forward security and basically more efficient
  - Update secret keys by using update ciphertexts which any sender can generate
  - Consider the security when a sender is an adversary

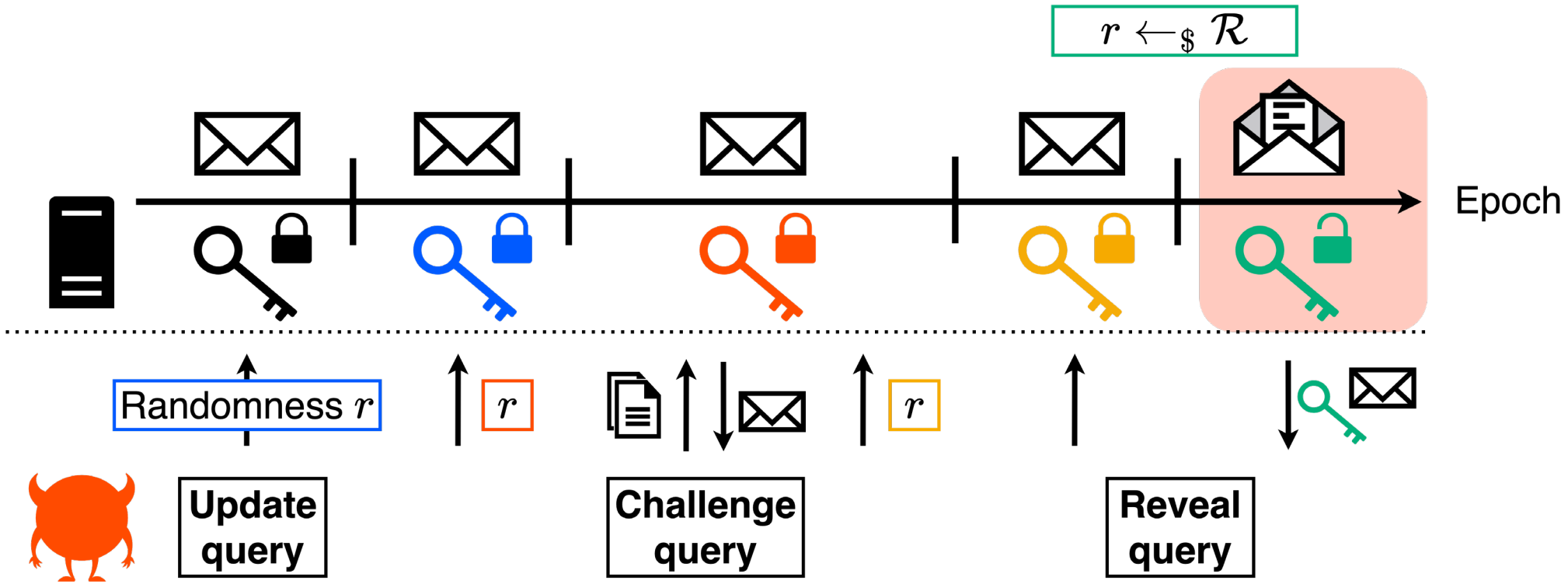





UEC IWAMOTO WATANABE LABORATORY Cryptography
RSAC 2025 Conference

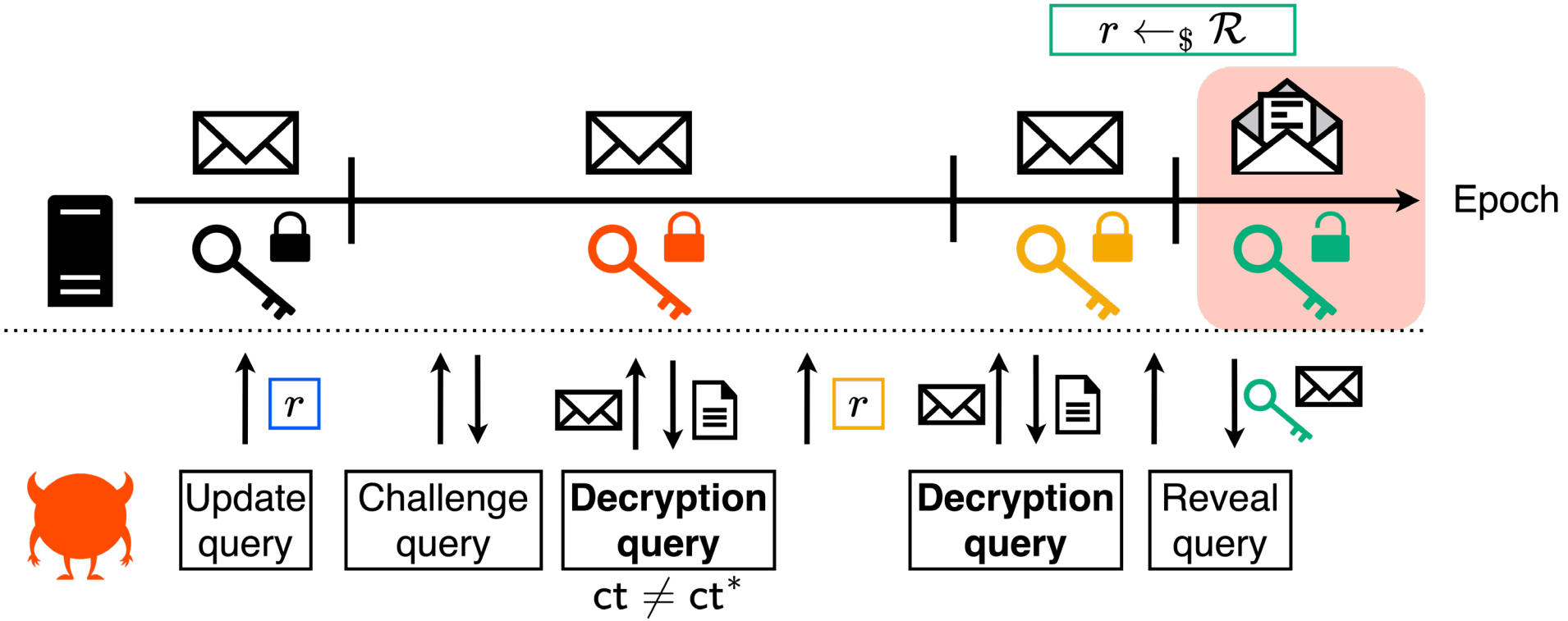
# UPKE: IND-CR-CPA Security

- **Update query**: The adversary can update keys with randomness of $r$ of their choosing
- **Reveal query**: A challenger updates keys and gives keys and an update ciphertext



# UPKE: IND-CR-CCA Security

- **Decryption query**: The adversary can obtain the decryption of a ciphertext $ct$
  - The adversary cannot make a decryption query on a challenge ciphertext $ct^*$ at the same epoch as made a challenge query



# Comparison Among Existing UPKE schemes (Random Oracle Model)

- IND-CR-CPA secure scheme
  - CDH assumption [JMM19, ACD+20]
- IND-CR/CU-CCA secure scheme
  - DCR assumption [HLP22], Extended-LWE assumption [HPS23], M-SIDH [DFV23]
  - Co-CDH w/ AGM [AFM24]

| Scheme | \|sk\| | \|pk\| | \|ct\| | \|up\| | Security | ROM | NIZK | AGM |
|---|---|---|---|---|---|---|---|---|
| [JMM19] | **256b** | **512b** | **512b** | **512b** | IND-CR-CPA | yes | no | no |
| [HLP22] | 4.6kb | 6.1kb | 12kb | 12kb | IND-CR-CPA | no | no | no |
| [HLP22] | 4.6kb | 6.1kb | 66kb | 12kb | **IND-CR-CCA** | yes | yes | no |
| [AFM24] | **384b** | **384b** | **768b** | **768b** | **IND-CU-CCA** | yes | yes | yes |

Comparison of the sizes of parameters for existing UPKE schemes with 128-bit security

# Comparison Among Existing UPKE schemes (Standard Model)

- IND-CR-CPA secure scheme
  - DCR assumption [HLP22], DDH and LWE assumption [DKW21], Extended-LWE assumption [HPS23], M-SIDH [DFV23]
- IND-CR-CCA secure scheme
  - Generic construction [DKW21]

- Dodis et al.'s construction relies on special NIZK arguments
  - It is unclear how to instantiate in practical and efficient way

**Compared to most efficient Jost et al.'s CPA-secure scheme, there is a big difference in efficiency**

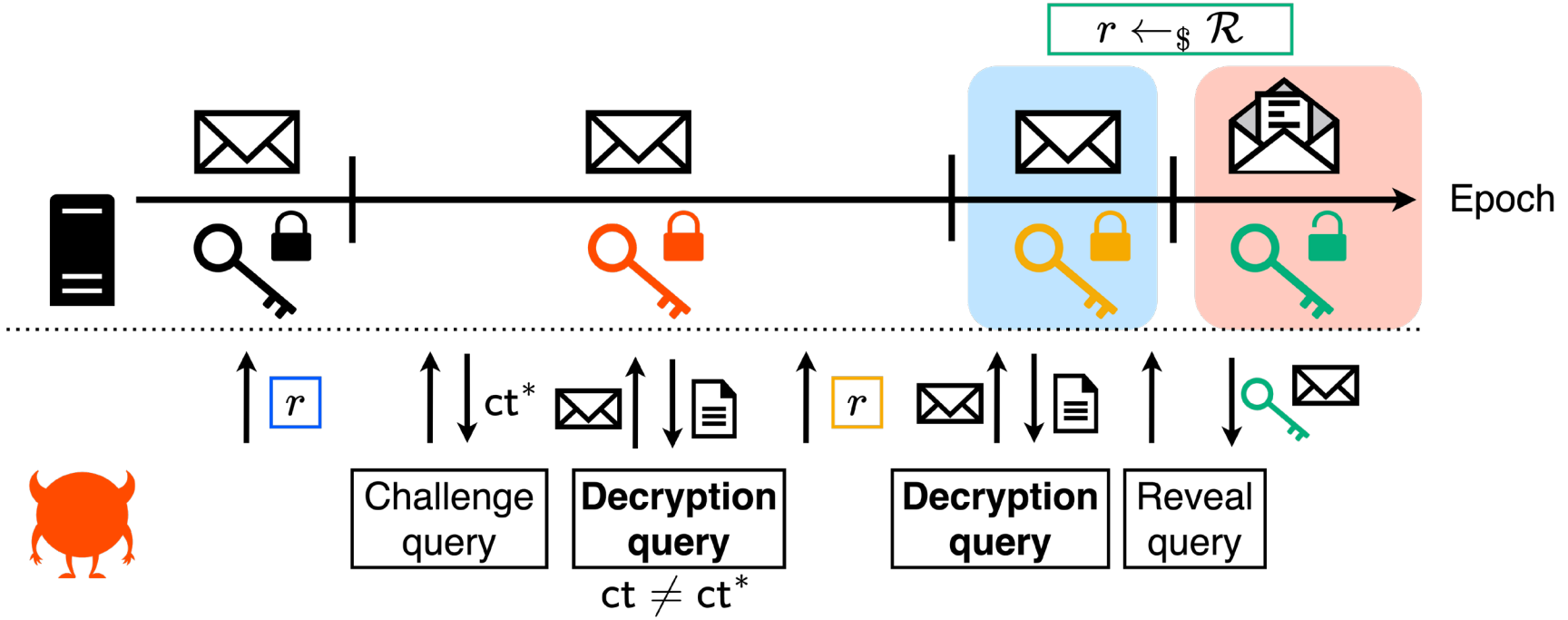
# Our Contribution

Chosen Update (CU) without public verifiability

- Propose a generic construction of CR/CU*-CCA-secure UPKE from CPA-secure UPKE
  - Construction is based on the Fujisaki–Okamoto transformation [FO99, FO13]
  - We can obtain the most efficient CCA-secure UPKE schemes
- Show that Dodis et al.'s construction [DKW21] is not CCA secure (in many cases)
  - It inherits the property that the underlying CPA-secure scheme has

| Scheme | \|sk\| | \|pk\| | \|ct\| | \|up\| | Security | ROM | NIZK | AGM |
|---|---|---|---|---|---|---|---|---|
| [JMM19] | **256b** | **512b** | **512b** | **512b** | IND-CR-CPA | yes | no | no |
| [HLP22] | 4.6kb | 6.1kb | 12kb | 12kb | IND-CR-CPA | no | no | no |
| [HLP22] | 4.6kb | 6.1kb | 66kb | 12kb | **IND-CR-CCA** | yes | yes | no |
| [AFM24] | **384b** | **384b** | **768b** | **768b** | **IND-CU-CCA** | yes | yes | yes |
| **Ours** w/ [JMM19] | **256b** | **640b** | **768b** | **512b** | **IND-CR-CCA** | yes | no | no |
| **Ours** w/ [JMM19] | **256b** | **640b** | **768b** | **1kb** | **IND-CU*-CCA** | yes | no | no |

# Requirements for IND-CR-CCA security




- Can make a decryption query on $ct^*$ from the next epoch after a challenge query
- A Ciphertext in any epoch must not be decryptable in another epoch
  - Otherwise, an adversary can break the security by making a decryption query on $ct^*$ at another epoch

# Dodis et al.'s Construction [DKW21]: Overview

- The attack success if we can make an update query which $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1})$
- The construction inherits the property that the underlying UPKE scheme has
  - We can break the security if there exists a UPKE scheme that such property has

IND-CR-CPA secure UPKE + NIKZ ⇒ IND-CR-CCA secure UPKE

$\widehat{\mathsf{Enc}}(\widehat{\mathsf{pk}_i} = (\mathsf{pk}_i, \mathsf{CRS}), \mathsf{M}) \to \widehat{\mathsf{ct}} = (\mathsf{ct}, \pi)$

1. $r \leftarrow_{\$} \mathcal{R}$
2. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{M}; r)$
3. $\pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, (\mathsf{pk}_i, \mathsf{ct}), (\mathsf{M}, r))$

$\widehat{\mathsf{Dec}}(\widehat{\mathsf{pk}_i}, \widehat{\mathsf{sk}_i} = \mathsf{sk}_i, \widehat{\mathsf{ct}}) \to \mathsf{M}$ or $\perp$

1. if $\mathsf{Vrfy}((\mathsf{pk}_i, \mathsf{ct}), \pi) = 0$, outputs $\perp$
   - **Does not check an epoch**
2. Output $\mathsf{Dec}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ct})$

- CRS: Common Reference String
- $\pi$: A proof that $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{M}; r)$ holds

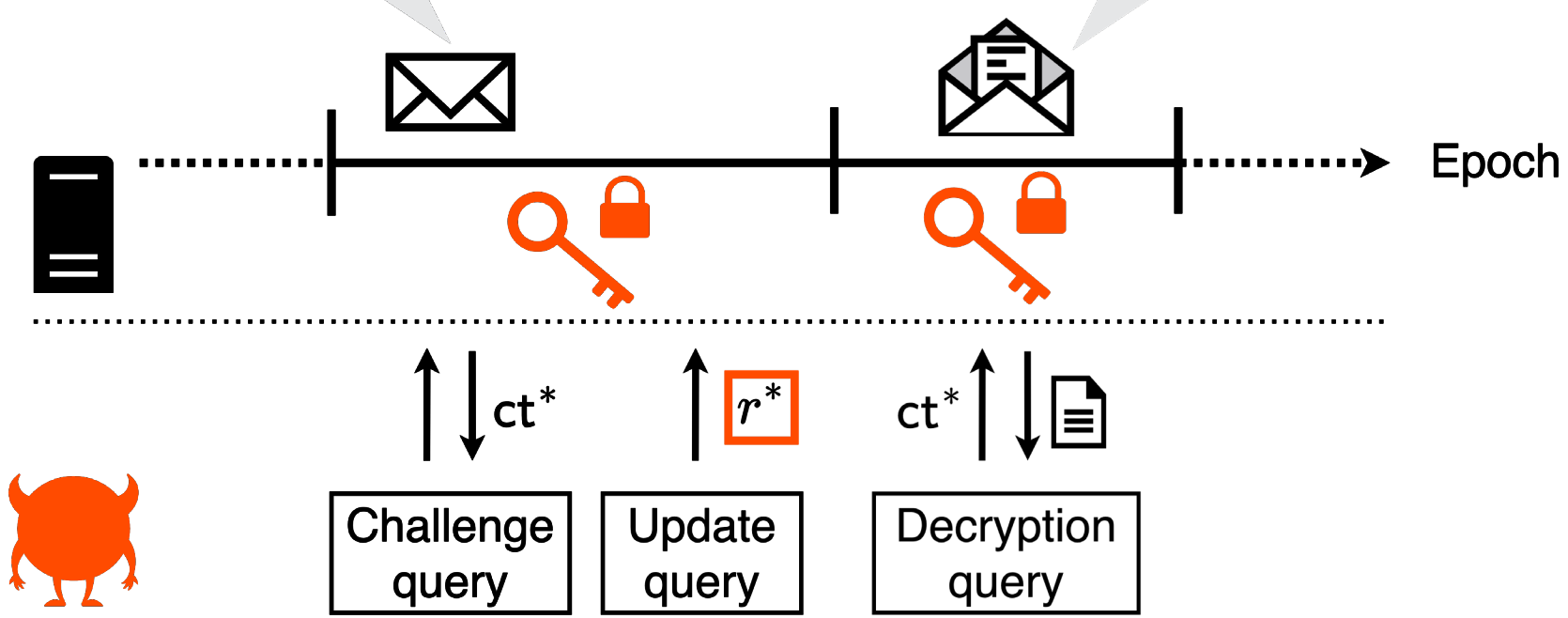
# Dodis et al.'s Construction [DKW21]: How to Attack

$\widehat{\mathsf{Enc}}(\widehat{\mathsf{pk}}_i = (\mathsf{pk}_i, \mathsf{CRS}), \mathsf{M}) \to \widehat{\mathsf{ct}} = (\mathsf{ct}, \pi)$

1. $r \leftarrow_{\$} \mathcal{R}$
2. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{M}; r)$
3. $\pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, (\mathsf{pk}_i, \mathsf{ct}), (\mathsf{M}, r))$

$\widehat{\mathsf{Dec}}(\widehat{\mathsf{pk}}_i, \widehat{\mathsf{sk}}_i = \mathsf{sk}_i, \widehat{\mathsf{ct}}) \to \mathsf{M}$ or $\perp$

1. If $\mathsf{Vrfy}((\mathsf{pk}_i, \mathsf{ct}), \pi) = 0$, outputs $\perp$
   - **Does not check an epoch**
2. Output $\mathsf{Dec}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ct})$



# Dodis et al.'s Construction [DKW21]: How to Fix

- In all existing CPA-secure UPKE schemes, an update query can be made which $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1})$ holds

→ Resolve this problem by embedding an epoch in a ciphertext

- Let $\widehat{\mathsf{pk}}_i := (\mathsf{pk}_i, i)$, then $\widehat{\mathsf{pk}}_i \neq \widehat{\mathsf{pk}}_j$ holds

$\widehat{\mathsf{Enc}}(\widehat{\mathsf{pk}}_i = (\mathsf{pk}_i, \mathsf{CRS}), \mathsf{M}) \to \widehat{\mathsf{ct}} = (\mathsf{ct}, \pi)$

1. $r \leftarrow_{\$} \mathcal{R}$
2. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{M}; r)$
3. $\pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, (\mathsf{pk}_i, \mathsf{ct}, i), (\mathsf{M}, r))$

$\widehat{\mathsf{Dec}}(\widehat{\mathsf{pk}}_i, \widehat{\mathsf{sk}}_i = \mathsf{sk}_i, \widehat{\mathsf{ct}}) \to \mathsf{M}$ or $\perp$

1. If $\mathsf{Vrfy}((\mathsf{pk}_i, \mathsf{ct}, i), \pi) = 0$, outputs $\perp$
2. Output $\mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ct})$

- CRS: Common Reference String
- $\pi$: A proof that $(\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}_i, \mathsf{M}; r)) \wedge (i = j)$ holds

- We can also fix by just concatnating a message and epoch

# Our Construction: Based on the Fujisaki–Okamoto transformation [FO99, FO13]

- In all existing CPA-secure UPKE schemes, an update query can be made which $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1})$ holds

→ Resolve this problem by embedding an epoch in a ciphertext

- Let $\widehat{\mathsf{pk}}_i := (\mathsf{pk}_i, i)$, then $\widehat{\mathsf{pk}}_i \neq \widehat{\mathsf{pk}}_j$ holds

$\widehat{\mathsf{Enc}}(\widehat{\mathsf{pk}}_i = (\mathsf{pk}_i, \mathsf{CRS}), \mathsf{M}) \to \widehat{\mathsf{ct}} = (\mathsf{ct}, \pi)$

1. $\sigma \leftarrow_{\$} \mathcal{M}_{\mathsf{asy}}$
2. $k = \mathsf{G}(\sigma)$
3. $\mathsf{ct}_{\mathsf{asy}} \leftarrow \mathsf{SKE.Enc}(k, \mathsf{M})$
4. $h = \mathsf{H}(i, \sigma, \mathsf{ct}_{\mathsf{asy}})$
5. $\mathsf{ct}_{\mathsf{asy}} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \sigma; h)$

- $\mathsf{G}\colon \{0,1\}^* \to \mathcal{SK}_{\mathsf{asy}}$
- $\mathsf{H}\colon \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \mathcal{R}$

# Conclusion

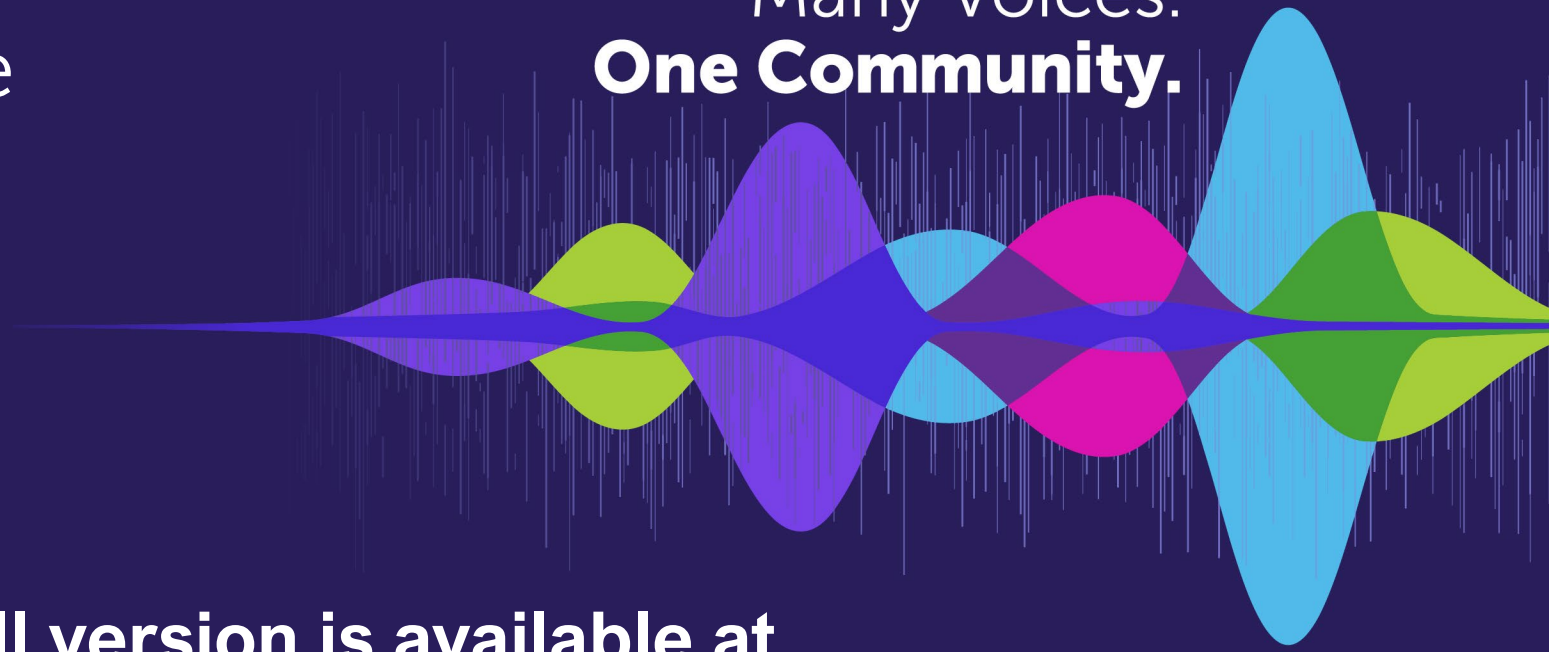
Chosen Update without public verifiability

- Propose a generic construction of CR/CU*-CCA-secure UPKE from CPA-secure UPKE
  - Construction is based on the Fujisaki–Okamoto transformation [FO99, FO13]
  - We can obtain the most efficient CCA-secure UPKE schemes
- Show that Dodis et al.'s construction [DKW21] is not CCA secure (in many cases)
  - It inrehits the property that the underlying CPA-secure scheme has

| Scheme | \|sk\| | \|pk\| | \|ct\| | \|up\| | Security | ROM | NIZK | AGM |
|---|---|---|---|---|---|---|---|---|
| [JMM19] | **256b** | **512b** | **512b** | **512b** | IND-CR-CPA | yes | no | no |
| [HLP22] | 4.6kb | 6.1kb | 12kb | 12kb | IND-CR-CPA | no | no | no |
| [HLP22] | 4.6kb | 6.1kb | 66kb | 12kb | **IND-CR-CCA** | yes | yes | no |
| [AFM24] | **384b** | **384b** | **768b** | **768b** | **IND-CU-CCA** | yes | yes | yes |
| **Ours** w/ [JMM19] | **256b** | **640b** | **768b** | **512b** | **IND-CR-CCA** | yes | no | no |
| **Ours** w/ [JMM19] | **256b** | **640b** | **768b** | **1kb** | **IND-CU*-CCA** | yes | no | no |

RSAC™ 2025 Conference

Many Voices.
**One Community.**



**The full version is available at**
**https://eprint.iacr.org/2023/976**